Stochastic Gradient Descent

STAT 672 Project

Tom Wallace

George Mason University

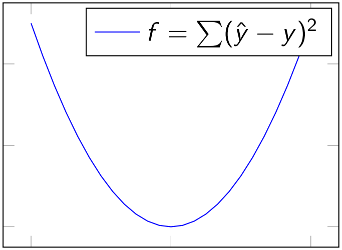
Spring 2018

Optimization is everywhere, and sometimes is easy

Many statistical procedures involve minimizing or maximizing some function applied to data

In **parametric** statistics, we often make assumptions that make this optimization "nice":

- Example: in OLS, we do not need to try different values of $\hat{\beta}$ to see which minimizes the loss function, we (typically) can just evaluate $(X'X)^{-1}X'Y$



Other times, optimization is not so easy

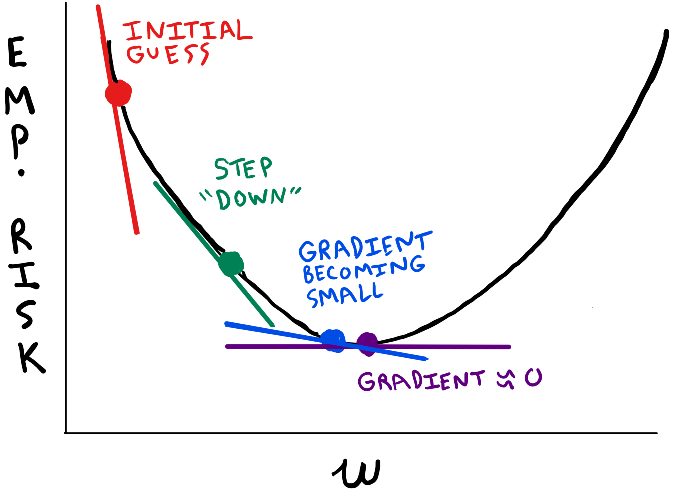
Suppose that we have a typical supervised classification problem:

- Non-parametric: no assumptions about distribution of data
- Feature vector $\mathbf{X}_i$, label Y_i
- Want to find best prediction function f_w^* from class $\mathcal{F}$
- Optimization: pick weights $\mathbf{w}$ that minimize empirical risk according to some convex loss function $L(f_w(x), y)$

Our lack of assumptions requires a different approach to optimization

- Cannot analytically identify stationary point
- Need to numerically search for it

Gradient descent is an iterative search procedure



A more formal explanation

$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma \frac{1}{n} \sum_{i=1}^{n} \nabla_w L(f_w(\mathbf{X}_i), y_i)$

Stop if $\mathbf{w}_t - \ldots \leq \epsilon$

Step size γ can vary over time

Theoretical guarantees on speed of convergence ($\rho :=$ size of error):

- Version presented here: $-\log \rho \sim t$
- More optimized version: $-\log \log \rho \sim t$

But, big difference between:

- Speed := number of iterations
- Speed := time (clock on wall)

Batch gradient descent is computationally expensive

In "plain" (batch) gradient descent, for every step, we have to evaluate the gradient at every observation

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma \frac{1}{n} \sum_{i=1}^{n} \nabla_w L(f_w(\mathbf{X}_i), y_i)$$

This becomes computationally intractable as n grows large

Knowing that the quality of our approximation gets linearly or quadratically better with t is not comforting if each t takes days to run

Stochastic gradient descent (SGD) takes less time

For each step, gradient is computed for a **single** randomly chosen observation *i*:

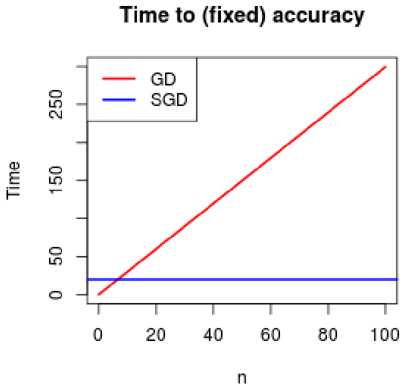
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma \nabla_w L(f_w(\mathbf{X}_i), y_i)$$

This simplification makes approximation much “noisier”, and hence SGD requires more iterations

But, each iteration is faster and so SGD can reach a predefined level of risk or error in less time

SGD is particularly useful when n is large and computation time is important

	GD	**SGD**
Time to accuracy ρ	$n \log \frac{1}{\rho}$	$\frac{1}{\rho}$



SGD is widely used in industry

If a Silicon Valley press release uses any of the following phrases...

- "Neural networks"
- "Machine learning"
- "AI"

...SGD probably is involved. Example: Google's **AlphaGo** program.

nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO